جلسه‌ی دوم

فصل ۱ و ۲ کتاب CLRS را بخوانید.

واژه‌نامه‌ی داده‌ساختارها و الگوریتم‌ها: `http://ce.sharif.edu/~dic-ads`

روش‌های تحلیل الگوریتم‌ها

هدف‌های تحلیل الگوریتم‌ها به شرح زیر است:

- بررسی رفتار الگوریتم قبل از پیاده‌سازی، از نظر زمان اجرا و مقدار حافظه‌ی مصرفی
- مقایسه‌ی الگوریتم‌ها از نظر کارایی

زمان اجرای الگوریتم‌ها

عوامل زیر در زمان اجرای یک برنامه موثرند:

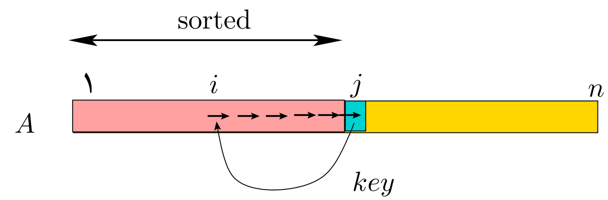
۱. سرعت سخت افزار
۲. نوع کامپایلر
۳. اندازه‌ی داده‌ی ورودی مسئله
۴. ترکیب داده‌های ورودی
۵. پیچیدگی الگوریتم
۶. پارامترهای دیگر که تاثیر ثابت در زمان اجرا دارند

زمان اجرای الگوریتم: $T(n)$، n اندازه‌ی ورودی مسئله

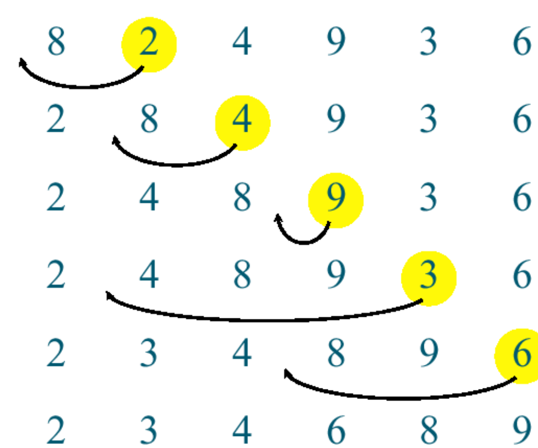
توجه:

- ممکن است چند داده‌ی ورودی داشته باشیم:
- یک گراف، تعداد راس‌ها= n، تعداد یال‌ها= m، $T(n,m)$
- چند پارامتر، انتزاع (abstraction)

مثال: مرتب‌ساز درجی (Insertion-Sort)




```
NSERTION-SORT(A, n)
     ▷ A: (the array to sort)
     ▷ n: (size of array)
1  for k ← 2 to n
2      do key ← A[k]
3          i ← k − 1
4          while i > 0 and A[i] > key
5              do A[i + 1] ← A[i]
6                  i ← i − 1
7          A[i + 1] ← key
```

سطر	هزینه	تعداد
۱	$c_۱$	n
۲	$c_۲$	$n-۱$
۳	$c_۳$	$n-۱$
۴	$c_۴$	$\sum_{k=۲}^{n} t_k$
۵	$c_۵$	$\sum_{k=۲}^{n} (t_k-۱)$
۶	$c_۶$	$\sum_{k=۲}^{n} (t_k-۱)$
۷	$c_۷$	$n-۱$

$$\begin{aligned} T(n) &= c_1 n + (c_2 + c_3 + c_7)(n-1) + c_4 \sum_{k=2}^{n} t_k + \\ & (c_5 + c_6) \sum_{k=2}^{n} (t_k - 1) \\ &= An + B \sum_{k=2}^{n} t_k + C \end{aligned}$$

بدترین حالت

در بدترین حالت (worst-case):

حداکثر مقدار t_k برابر k است و این زمانی است که آرایه برعکس مرتب شده باشد.

$$\begin{aligned} T(n) &= An + B \sum_{k=2}^{n} t_k + C \\ &= An + B\frac{(n+1)(n+2)}{2} + C \\ &= an^2 + bn + c \end{aligned}$$

به‌ترین حالت

به‌ترین حالت (best-case)**: آرایه از قبل مرتب باشد. در این حالت داریم $t_k = 1$ و**

$$T(n) = An + B(n-1) + C$$

که بر حسب n خطی است و با بدترین حالت که درجه دو است تفاوت دارد.

پس مسئله‌ی یافتن حالت متوسط (average-case) **مطرح می‌شود.**

حالت متوسط

داده‌ی ورودی به صورت تصادفی داده است.

فرض می‌شود که همه‌ی حالت‌های مختلف داده‌های ورودی، احتمال برابر دارند ($\frac{1}{n!}$)

حالت متوسط در مورد مثال فوق برابر است با

$$\overline{T}(n) = An + C + \frac{1}{n!} \sum_{i=1}^{n!} \sum_{k=2}^{n} B t_{k,i}$$

$$\frac{1}{n!} \sum_{i=1}^{n!} t_{k,i} = \overline{t}_k.$$

یعنی

$$\overline{T}(n) = An + C + B\sum_{k=۲}^{n}\bar{t}_k.$$

$$\overline{T}(n) = An + C + B\sum_{k=۲}^{n}\bar{t}_k$$

i: **مکانی را که عضو باید در آن جا بگیرد** ($۱ \le i \le k$). $\Longleftarrow$ $t_k = k - i + ۱$

$$\begin{aligned}\bar{t}_k &= \sum_{i=۱}^{k}\frac{۱}{k}(k-i+۱)\\ &= \frac{۱}{k}\times\frac{k(k+۱)}{۲}\\ &= \frac{k+۱}{۲}\end{aligned}$$

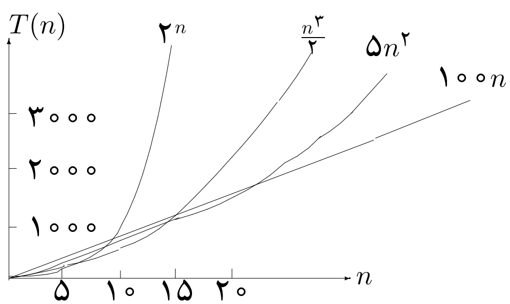
$$\begin{aligned}\overline{T}(n) &= An + C + B\sum_{k=۲}^{n}\frac{k+۱}{۲}\\ &= an^{۲} + bn + c\end{aligned}$$

پس رفتار حالت متوسط و بدترین حالت یکسان است.

مرتبه الگوریتم‌ها

پیچیدگی الگوریتم‌ها (complexity of algorithms)

الگوریتم	$T(n)$	$O(.)$	حداکثر اندازه‌ی مسئله، قابل حل در ۱۰۰۰ ثانیه	حداکثراندازه برای ماشین ۱۰ برابر سریع‌تر	نسبت	نسبت برای ماشین ۱۰۰۰ برابر سریع‌تر
$A_۱$	$۱۰۰n$	$O(n)$	۱۰	۱۰۰	۱۰	۱۰۰۰
$A_۲$	$۵n^۲$	$O(n^۲)$	۱۴	۴۵	۳/۲	۳۱/۹۴
$A_۳$	$n^۳/۲$	$O(n^۳)$	۱۲	۲۷	۲/۳	۱۲۵/۹۹
$A_۴$	۲^n	$O(۲^n)$	۱۰	۱۳	۱/۳	۲



زمان‌های اجرای چهار الگوریتم برای یک مسئله.

تابع‌های رشد (growth functions)

سه نماد O، Θ و Ω

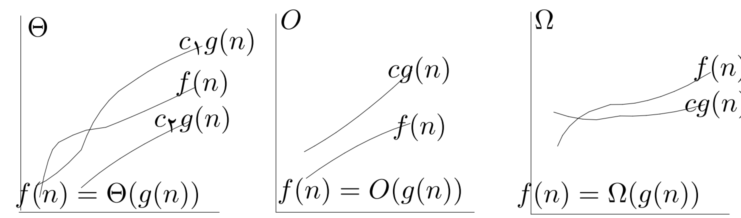
می‌گوییم

- بدترین حالت الگوریتم مرتب‌ساز درجی، $T(n) = \Theta(n^2)$ یعنی از مرتبه‌ی دقیق n^2 است. در حالت متوسط نیز $\Theta(n^2)$ است. است.
- الگوریتم مرتب‌ساز هیپ (heap sort) از $O(n \log n)$ یا از مرتبه‌ی $n \log n$ است.
- کلیه‌ی الگوریتم‌های مرتب‌ساز مبتنی بر مقایسه، از مرتبه‌ی $\Omega(n \log n)$ هستند.

تعریف

$$\Theta(g(n)) = \{f(n) : \exists c_1, c_2 > 0 \text{ and } n_0 \text{ such that } \forall n \geq n_0,\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

عبارت $c_1 g(n) \leq f(n) \leq c_2 g(n)$ به این معنی است که برای مقادیر بزرگ n درجه‌ی رشد f و g یکسان است.



منحنی‌های رشد.

$$f(n) \in \Theta(g(n))$$

و می‌گوییم که تابع $g(n)$ کران بالای بسته‌ی مجانبی (asymptotically tight bound برای $f(n)$ است.

به‌شکل ساده‌تر

$$f(n) = \Theta(g(n))$$

در مورد الگوریتم مرتب‌ساز مبتنی بر درج، $T(n) = \Theta(n^2)$. زیرا می‌توان مقادیر مثبتی برای c_1 و c_2 را طوری یافت که

$$c_1 n^2 \leq an^2 + bn + c \leq c_2 n^2$$

مسئله: ثابت کنید که $100n^2 + 5n - 4 = \Theta(n^2)$.

ثابت کنید که $100n^2 + 5n - 4 = \Theta(n^2)$.

حل:

روشن است که اگر $c_1 = 1$ و $c_2 = 200$، برای همه‌ی مقادیر $n > 1$ داری
$c_1 n^2 \leq 100n^2 + 5n - 4 \leq c_2 n^2$.

مسئله: نشان دهید که $100n^2 + 5n - 4 \neq \Theta(n^3)$.

حل:

باید نشان دهیم که هیچ مقادیر مثبت برای c_1 و c_2 و یک مقدار برای n_0 پیدا نمی‌شود که رابطه‌ی $c_1 n^3 \leq 100n^2 + 5n - 4 \leq c_2 n^3$ برای همه‌ی مقادیر $n > n_0$ برقرار باشد.
به‌وضوح به‌ازای هر مقدار $c_1 > 0$ و برای nهای بزرگ داریم $c_1 n^3 \not\leq 100n^2 + 5n - 4$.

می‌توان ثابت کرد که هر چندجمله‌ای از مرتبه‌ی دقیق جمله‌ی با بزرگ‌ترین توانش است یعنی

$$a_k n^k + a_{k-1} n^{k-1} + \cdots = \Theta(n^k)$$

در مورد مرتب‌ساز درجی داریم:

$$\begin{aligned} T(n) &= \Theta(n^2) \\ &= \Theta(100n^2) \\ &\neq \Theta(n^3) \\ &\neq \Theta(n^2 \log n) \\ &\neq \Theta(n \log n) \end{aligned}$$

برای اثبات $T(n) \neq \Theta(n \lg n)$ می‌توان نشان داد که نمی‌توان ثابت‌های c_1 و c_2 را پیدا کرد که برای $n > n_0$ داشته‌باشیم:

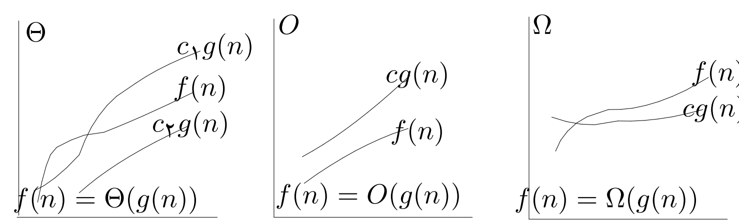
$$c_1 n \lg n \leq n^2 \leq c_2 n \lg n$$

نماد O

برای $g(n)$ داده‌شده، $O(g(n))$ را به شکل مجموعه‌ی توابع زیر تعریف می‌کنیم:

$$O(g(n)) = \{f(n) : \exists c > 0 \text{ and } n_0 \text{ such that } \forall n \geq n_0\ 0 \leq f(n) \leq cg(n)\}$$

$g(n)$ **کران بالای مجانبی** (asymptotically upper bound) **برای** $f(n)$ **است.**



منحنی‌های رشد.

مثلاً در مورد مرتب‌ساز درجی، داریم

$$\begin{aligned} T(n) = \Theta(n^2) &= O(n^2) \\ &= O(100n^2) \\ &= O(n^3) \\ &= O(n^2 \lg n) \\ &\neq O(n \lg n) \\ &\neq O(n). \end{aligned}$$

می‌توان گفت مسئله از $O(n^{100})$ است، ولی این اطلاع چندان مفید نیست. به‌همین جهت در حالت‌هایی که محاسبه‌ی Θ مشکل است، باید تابع داخل پرانتز O را تا حد امکان کوچک‌تر باشد.

نکته: براساس تعریف روشن است که $\Theta(g(n)) \subseteq O(g(n))$.

نماد Ω

برای $g(n)$ داده‌شده، $\Omega(g(n))$ را به‌شکل مجموعه‌ی توابع زیر تعریف می‌کنیم:

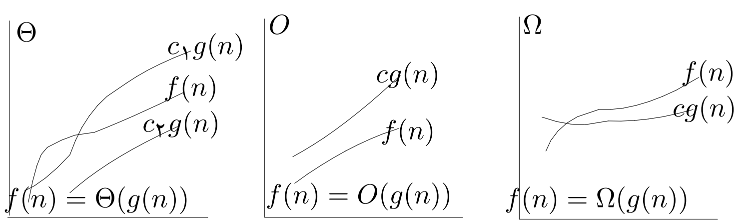
$$\Omega(g(n)) = \{f(n) : \exists c > 0 \text{ and } n_0 \text{such that } \forall n \geq n_0 \; 0 \leq cg(n) \leq f(n)\}$$

در این صورت، $g(n)$ کران پایین مجانبی (asymptotically lower bound**) برای $f(n)$ است.**

مثلاً برای مرتب‌ساز درجی داریم

$$\begin{aligned} T(n) &= \Omega(n \log n) \\ &= \Omega(n) \\ &= \Omega(n^2) \\ &\neq \Omega(n^2 \log n) \end{aligned}$$

در عمل نماد Ω برای نشان دادن حد پایین یک تابع دیگر به‌کار می‌رود.



منحنی‌های رشد.

تابع Θ در واقع عطف O و Ω است یعنی

$$f(n)=\Theta(g(n)) \Leftrightarrow f(n)=O(g(n)) \wedge f(n)=\Omega(g(n))$$

قضیه: شرط لازم و کافی برای $f(n)=\Theta(g(n)$ **آن است که** $f(n)=O(g(n)$
.$f(n)=\Omega(g(n)$

نمادهای o و w

$$o(g(n))=\{f(n)| \quad \text{for any positive constant} \quad c>0, \quad \exists n_0>0, s.t. \forall n>n_0: 0\leq f(n)<cg(n)\}$$

این رابطه نزدیکی زیادی با O دارد بااین تفاوت که دیگر $f(n)$ نمی‌تواند با $cg(n)$ برابر باشد و باید الزاماً کوچک‌تر باشد.

نکته: اگر $f(n)=o(g(n))$، **داریم:** $\lim_{n\to\infty}\frac{f(n)}{g(n)}=0$.

$$\omega(g(n))=\{g(n)| \quad \text{for any positive constant} \quad c>0, \exists n_0>0, \quad \text{such that} \quad 0\leq cg(n)<F(n)\}$$

رابطه‌ی ω با Ω مثل o با O است.

نکته: $f(n)=o(g(n))$ **اگر و فقط اگر** $g(n)=\omega(f(n))$.

نکته: اگر $f(n)=\omega(g(n))$**، داریم:** $\lim_{n\to\infty}\frac{f(n)}{g(n)}=\infty$.

مثلاً $n^2/2=\omega(n)$ **ولی** $n^2/2\neq\omega(n^2)$.

نتیجه‌ای که از تعریف‌های فوق به‌دست می‌آید را در عبارت‌های زیر (هرچند نادقیق ا نظر ریاضی) خلاصه می‌کنیم:

اگر F **درجه‌ی رشد** f **و** G **درجه‌ی رشد** g **برای اندازه‌ی ورودی زیاد باشد،**

$$f=\Theta(g)\iff F=G$$
$$f=O(g)\iff F\leq G$$
$$f=o(g)\iff F<G$$
$$f=\Omega(g)\iff F\geq G$$
$$f=\omega(g)\iff F>G$$

خواص تابع‌های رشد

خاصیت تراگذاری

۱) از $f(n)=\Theta(g(n))$ **و** $g(n)=\Theta(h(n))$ **نتیجه می‌شود که** $f(n)=\Theta(h(n))$.

۲) از $f(n)=O(g(n))$ **و** $g(n)=O(h(n))$ **نتیجه می‌شود که** $f(n)=O(h(n))$.

۳) از $f(n)=\Omega(g(n))$ **و** $g(n)=\Omega(h(n))$ **نتیجه می‌شود که** $f(n)=\Omega(h(n))$.

۴) از $f(n)=o(g(n))$ **و** $g(n)=o(h(n))$ **نتیجه می‌شود که** $f(n)=o(h(n))$.

۵) از $f(n)=\omega(g(n))$ **و** $g(n)=\omega(h(n))$ **نتیجه می‌شود که** $f(n)=\omega(h(n))$.

خاصیت انعکاسی

۱) $f(n)=\Theta(f(n))$.

۲) $f(n)=O(f(n))$.

۳) $f(n)=\Omega(f(n))$.

خاصیت تقارن

$f(n)=\Theta(g(n))$ **اگر و فقط اگر** $g(n)=\Theta(f(n))$.

خاصیت تقارن جابه‌جایی

۱) شرط لازم و کافی برای $f(n)=O(g(n))$ **آن است که** $g(n)=\Omega(f(n))$.

۲) شرط لازم و کافی برای $f(n)=o(g(n))$ **آن است که** $g(n)=\omega(f(n))$.

تمرین: همین‌جا حل کنید!

۱) آیا $f(n)+g(n)=\Theta(\min(f(n),g(n)))$ **است؟**

۲) آیا $f(n)=\Theta(f(n)/۲)$ **است؟**

۳) آیا $f(n)+o(f(n))=\Theta(f(n)$؟

روش‌های تحلیل الگوریتم‌ها

الگوریتم‌های ترتیبی

- **یک یا تعداد ثابتی عبارت ساده** (Statement) $\Leftarrow \Theta(۱)$

- $T(n)$ **زمان اجرای یک تکه‌برنامه که به‌صورت زیر است:**
  - ◁ **چند تکه‌برنامه با زمان‌های اجرای**

$$\begin{aligned} T_1(n) &= O(f_1(n)) \\ T_2(n) &= O(f_2(n)) \\ &\cdots \\ T_k(n) &= O(f_k(n)) \end{aligned}$$

$$T(n) = \sum_{i=1}^{k} T_i(n) = O(\max_{1\le i\le k}(f_i(n)))$$

◁ **تکرار** $g(n)$ **بار تکه‌برنامه‌ای با زمان اجرای** $S(n) = O(f(n))$

$$T(n) = g(n)S(n) = O(g(n)f(n))$$

◁ **ساختار** if **باشد و قسمت‌های** then و else **آن به ترتیب زمان‌های** $T_1(n) = O(f_1(n))$ **و** $T_2(n) = O(f_2(n))$

$$T(n) = \max\{T_1(n), T_2(n)\} = O(\max\{f_1(n), f_2(n)\})$$

مثال: مرتب‌ساز حبابی

```
BUBBLE-SORT(A)
for i ← 1 to length[A] − 1
    do for j ← length[A] downto i + 1
        do if A[j] > A[j − 1]
            then SWAP (a[j], A[j − 1] )
```


مثال: مرتب‌ساز حبابی

```
BUBBLE-SORT(A)
for i ← 1 to length[A] − 1
    do for j ← length[A] downto i + 1
        do if A[j] > A[j − 1]
            then SWAP (a[j], A[j − 1] )
```

$$\begin{aligned} T(n) &= \sum_{i=1}^{n-1}\sum_{i+1}^{n} O(1) \\ &= \sum_{i=1}^{n-1} c(n-i) \\ &= c[n(n-1) - \frac{n(n-1)}{2}] = c\frac{n(n-1)}{2} = O(n^2) \end{aligned}$$

تمرین: همین‌جا حل کنید!

زمان اجرای الگوریتم‌های زیر را محاسبه کنید:

Mystry(n)

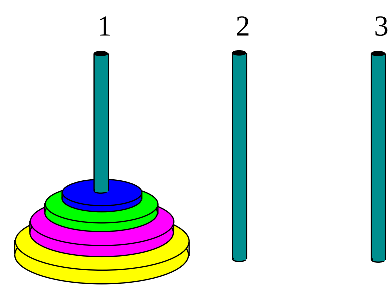
for $i \leftarrow 1$ **to** $n-1$
  do for $j \leftarrow i+1$ **to** n
    do for $k \leftarrow 1$ **to** j
      do some $O(1)$ statements

VeryOdd(n)

for $i \leftarrow 1$ **to** n
  do if $odd(i)$
    then for $j \leftarrow 1$ **to** n
      do $x \leftarrow x+1$
    for $k \leftarrow 1$ **to** i
      do $y \leftarrow y+1$

الگوریتم‌های بازگشتی

مثال: برج‌های هانوی



Tower-of-Honoi(n, f, t, h)

▷ moving n coins from leg f to leg t with the help of leg h

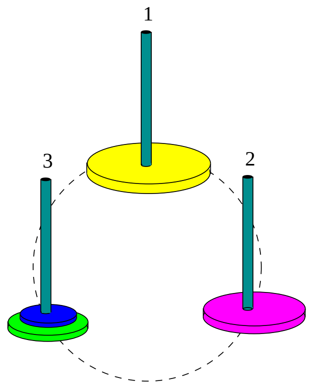
if $n = 1$
  then Move the coin from leg f to leg t
  else Tower-of-Honoi$(n-1, f, h, t)$
    Move the coin from leg f to leg t
    Tower-of-Honoi$(n-1, h, t, f)$

تعداد حرکت‌ها

$$T(n) = \begin{cases} ۱, & n = ۱ \\ T(n-۱) + ۱ + T(n-۱) & n > ۱ \end{cases}$$

یک رابطه‌ی بازگشتی (Recurrence Relation)

راه‌حل ترتیبی



تمرین

اگر در مسئله‌ی برج هانوی امکان انتقال سکه‌ها از میله‌ی ۱ به ۲ و یا از ۲ به ۱ نباشد، کم‌ترین تعداد انتقال n سکه از میله ۱ به ۲ چه‌قدر است؟

حل

$$T_{۱۲}(n) = \begin{cases} ۲, & n = ۱ \\ T_{۱۲}(n-۱) + ۱ + T_{۲۱}(n-۱) + ۱ + T_{۱۲}(n-۱) & n > ۱ \end{cases}$$

می‌دانیم $T_{۱۲} = T_{۲۱}$

اگر بخواهیم از ۱ به ۳ ببریم چه‌طور؟

حل

$$T_{۱۳}(n) = \begin{cases} ۱, & n = ۱ \\ T_{۱۲}(n-۱) + ۱ + T_{۲۳}(n-۱) & n > ۱ \end{cases}$$

می‌دانیم $T_{۲۳} = T_{۱۳}$

تمرین‌های دیگر

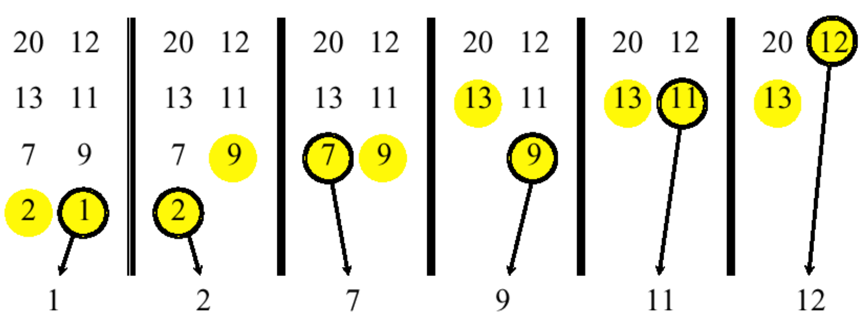
۱) اگر در برج هانوی، سکه‌های با شماره‌ی فرد و زوج به‌ترتیب در میله‌های ۱ و
باشند، با حداقل حرکت‌ها می‌خواهیم همه‌ی سکه‌ها را در میله‌ی ۳ قرار دهیم
چه‌گونه این کار را می‌شود انجام داد؟

۲) مسئله‌ی برج هانوی را اگر در ابتدا $n_۱$ سکه در میله‌ی ۱، $n_۲$ سکه در میله‌ی ۲
بقیه‌ی n سکه در میله‌ی سوم باشد را حل کنید به‌طوری که در انتها همه‌ی سکه‌ها د
میله‌ی سوم قرار گیرند. تعداد حرکت‌های بهینه را به‌دست آورید.

مثال: مرتب‌سازی ادغامی

```
MERGE-SORT(A, p, r)
    ▷ A: the array to sort
    ▷ p: starting index
    ▷ r: ending index
  if p < r
    then q ← ⌊(p+r)/2⌋
          MERGE-SORT(A, p, q)
          MERGE-SORT(A, q+1, r)
          MERGE (A, p, q, r)
```


ادغام دو آرایه‌ی مرتب: مثال



```
MERGE(A, B)
n ← length[A];   m ← length[B]
i ← 1;   j ← 1;   k ← 0
while i ≤ n or j ≤ m
      do k ← k + 1
         if j > m  or ((i ≤ n)  and  (A[i] ≤ B[j]))
            then C[k] ← A[i]
                 i ← i + 1
            else  C[k] ← B[j]
                 j ← j + 1
length[C] ← n + m
return C
```

تعداد مقایسه‌های عناصر همیشه $= n + m - ۱$


```
MERGE(A, B)
n ← length[A];   m ← length[B]
A[n + 1] ← ∞;   B[m + 1] ← ∞
i ← 1;   j ← 1
for k ← 1 to n + m
   do if A[i] < B[j]
        then C[k] ← A[i]
             i ← i + 1
        else  C[k] ← B[j]
             j ← j + 1
length[C] ← n + m
return C
```


ادغام دو لیست مرتب به طول‌های m و n در زمان $O(m+n)$ و فضای $O(m+n)$

ادغام درجا(in-place merging) از فضای اضافی $O(۱)$ استفاده می‌کند.

آیا مرتب‌سازی ادغامی درجاست؟

مرتب‌سازی ادغامی

اثبات درستی الگوریتم با استقرا

$$T(n) = \begin{cases} ۱ & n = ۲ \\ T(\lfloor n/۲ \rfloor) + T(\lceil n/۲ \rceil) + cn & n > ۲ \end{cases}$$

$$T(n) = \Theta(n \log n)$$

روش‌های حل رابطه‌های بازگشتی

۱) حدس و استقرا: حدس خوب و استقرا

۲) تکرار با جای‌گذاری: بازکردن فرمول و به‌دست آوردن جواب به‌طور صریح

۳) قضیه‌ی اصلی

۴) روش‌های حل رابطه‌های بازگشتی همگن و ناهمگن

۵) روش‌های دیگر مانند تابع مولد

حدس و استقرا: مثال

$$T(n) = ۲T(\lfloor \frac{n}{۲} \rfloor) + n$$
$$T(۲) = T(۱) = O(۱)$$

حدس و استقرا: مثال

$$T(n) = ۲T(\lfloor \frac{n}{۲} \rfloor) + n$$
$$T(۲) = T(۱) = O(۱)$$

حدس: (برای این‌که اثبات کنیم که $T(n) = O(n \lg n)$)
$T(n) \leq Cn \lg n$ برای یک عدد مثبت C

حدس و استقرا: مثال

$$T(n) = ۲T(\lfloor \frac{n}{۲} \rfloor) + n$$
$$T(۲) = T(۱) = O(۱)$$

پایه‌ی استقرا:

$$n = ۲ \Rightarrow T(۲) \leq ۲C \Rightarrow C \geq \frac{T(۲)}{۲} > ۰$$

فرض استقرا: برای $k < n$ فرض می‌کنیم $T(k) \leq Ck \lg k$

حکم استقرا: باید ثابت کنیم: $T(n) \le Cn\lg n$.

$$\begin{aligned} T(n) &\le 2C\lfloor \tfrac{n}{2} \rfloor \lg \lfloor \tfrac{n}{2} \rfloor + n \\ &\le Cn\lg\tfrac{n}{2} + n \\ &= Cn\lg n - Cn + n \\ &\le Cn\lg n \end{aligned}$$

اگر $C \ge 1$ حکم اثبات می‌شود.

اثبات $T(n) = \Theta(n\lg n)$

برای این‌کار باید اثبات کنیم که $T(n) \ge rn\lg n$ (برای یک مقدار مثبت r).

پایه‌ی استقرا:

$$n = 2 \Rightarrow T(2) \ge 2r \Rightarrow r \le \frac{T(2)}{2}$$

فرض استقرا: برای $k < n$ فرض می‌کنیم $T(k) \ge rk\lg k$

حکم استقرا: باید ثابت کنیم: $T(n) \le rn\log_2 n$

$$\begin{aligned} T(n) &\ge 2r\lfloor \tfrac{n}{2} \rfloor \lg \lfloor \tfrac{n}{2} \rfloor + n \\ &\ge 2r(\tfrac{n}{2} - 1)\lg\tfrac{n}{2} + n \\ &= rn\lg n - rn - 2r\lg\tfrac{n}{2} + n \\ &= rn\lg n + n - r(n + 2\lg\tfrac{n}{2}) \\ &\ge rn\lg n \end{aligned}$$

اگر $r \le \frac{1}{2}$ حکم اثبات می‌شود.

برای $r = \frac{1}{2}$ داریم $n - r(n + 2\lg\frac{n}{2}) = n - \lg\frac{n}{2} > 0$

مثال: مرتب‌سازی ادغامی

$$\begin{aligned} T(2) &= a \\ T(n) &= 2T(\frac{n}{2}) + bn \end{aligned}$$

مثال: مرتب‌سازی ادغامی

$$\begin{aligned} T(۱) = T(۲) &= a \\ T(n) &= ۲T(\frac{n}{۲}) + bn \end{aligned}$$

۱) حدس: $T(n) \leq cn\log n \Rightarrow T(n) = O(n\log n)$

۲) پایه: $T(۲) = a \leq ۲c$

۳) فرض استقرا: برای $\frac{n}{۲}$ داریم، $T(n/۲) <= c\frac{n}{۲}\log\frac{n}{۲}$

$$\begin{aligned} T(n) &\leq ۲c\frac{n}{۲}\log\frac{n}{۲} + bn \\ &= cn\log n + (b-c)n \leq cn\log n \\ &\leq cn\log n, \;\; if \;\; b-c \leq ۰ \;\; and \;\; c \geq b, c \geq a/۲ \end{aligned}$$

برای این که اثبات کنیم $T(n) = \Theta(n\lg n)$ **باید اثبات کنیم که** $T(n) = \Omega(n\lg n)$ **ه**
هست.

برای این کار باید اثبات کنیم که یک c **هست که برای** n **های بزرگ داریم**
$T(n) \geq cn\lg n$

پایه: $T(۲) = ۲ \geq ۲c$

فرض و حکم:

$$\begin{aligned} T(n) &\geq ۲c\frac{n}{۲}\log\frac{n}{۲} + bn \\ &= cn\log n + (b-c)n \leq cn\log n \\ &\geq cn\log n, \;\; if \;\; b-c \geq ۰ \;\; and \;\; c \leq a/۲ \end{aligned}$$

برای هر عدد دلخواه n **چکار کنیم؟**

$۲^k \leq n < ۲^{k+۱}$ **می توان اثبات کرد که مرتبه‌ی** $T(n)$ **همان مرتبه‌ی** $T(۲^k)$ **است.**

تشخیص حدس اشتباه

$$T(n) = T(\lfloor\frac{n}{۲}\rfloor) + T(\lceil\frac{n}{۲}\rceil) + ۱$$

تشخیص حدس اشتباه

$$T(n) = T(\lfloor \frac{n}{۲} \rfloor) + T(\lceil \frac{n}{۲} \rceil) + ۱$$

حدس: $T(n) = O(n)$. **باید ثابت کنیم** $T(n) \leq Cn$.

تشخیص حدس اشتباه

$$T(n) = T(\lfloor \frac{n}{۲} \rfloor) + T(\lceil \frac{n}{۲} \rceil) + ۱$$

حدس: $T(n) = O(n)$. **باید ثابت کنیم** $T(n) \leq Cn$.

$$T(n) \leq C\lfloor \frac{n}{۲} \rfloor + C\lceil \frac{n}{۲} \rceil + ۱ = Cn + ۱ \geq Cn$$

اشتباه!

حدس‌دیگر: $T(n) \leq Cn + B$.

$$T(n) \leq C\lfloor \frac{n}{۲} \rfloor + C\lceil \frac{n}{۲} \rceil + ۲B + ۱ = Cn + ۲B + ۱ \leq Cn + B \Rightarrow B < ۰$$

مثال دیگر

$$T(n) = ۲T(\lfloor \frac{n}{۲} \rfloor) + n$$

مثال دیگر

$$T(n) = ۲T(\lfloor\frac{n}{۲}\rfloor) + n$$

حدس: $T(n) \leq Cn$

مثال دیگر

$$T(n) = ۲T(\lfloor\frac{n}{۲}\rfloor) + n$$

حدس: $T(n) \leq Cn$

$$T(n) \leq ۲C\lfloor\frac{n}{۲}\rfloor + n \leq Cn + n = (C + ۱)n \geq Cn$$

اشتباه!

حدس جدید: $T(n) \leq Cn\log n + B$

درست است؟

مثال

$$T(n) = ۲T(\sqrt{n}) + \log_۲ n$$

مثال

$$T(n) = ۲T(\sqrt{n}) + \log_۲ n$$

متغیر جدید: $m = \log_۲ n \Rightarrow T(۲^m) = ۲T(۲^{\frac{m}{۲}}) + m$.

$$S(m) = ۲S(\frac{m}{۲}) + m$$

مثال

$$T(n) = ۲T(\sqrt{n}) + \log_۲ n$$

$$S(m) = O(m \log m) \Rightarrow T(n) = O(\log n \log \log n)$$

کوچک‌ترین و بزرگ‌ترین عناصر آرایه

پیدا کردن کوچک‌ترین و بزرگ‌ترین عناصر آرایه‌ی nتای A با کم‌ترین تعداد مقایسه‌ها

این مقدار $\lceil \frac{۳n}{۲} \rceil - ۲$ است. چرا؟

روش اول:

- **بزرگترینِ n عنصر با $n-1$ مقایسه**
- **کوچکترین عنصر با $n-2$ مقایسه**
- **در کل $2n-3$ مقایسه**

روش دوم:

- **با یک مقایسه بین هر دو عنصر متوالی Minها و Maxهای آن‌دو را پیدا می‌کنیم.**
- **کوچک‌ترین عنصر Minها و حداکثر یک عنصر اضافی**
- **بزرگ‌ترین عنصر Maxها و حداکثر یک عنصر اضافی**

بهینه برای $n = 2k$ با $n/2 + (n/2 - 1) + (n/2 - 1) = 3n/2 - 2$ مقایسه

برای $n = 2k + 1$ به تعداد $3k = \lfloor 3n/2 \rfloor$ مقایسه نیاز است. بهینه نیست!

روش سوم: (تقسیم و حل)

```
MaxMin(A, p, q)
      ▷ A: array
      ▷ p, q: the first and last indices
      ▷ will return the indices of both max and min
if p=q
  then max ← min ← A[p]
  else if q − p = 1
          then if A[p] > A[q]
                  then max ← A[p] ; min ← A[q]
                  else max ← A[q] ; min ← A[p]
          else r ← ⌊(p+q)/2⌋
               min1, max1 ← MaxMin(A, p, r)
               min2, max2 ← MaxMin(A, r + 1, q)
               if max2 > max1
                  then max ← max2
                  else max ← max1
               if min2 < min1
                  then min ← min2
                  else min ← min1
```


$$T(n) = \begin{cases} 0 & n = 1 \\ 1 & n = 2 \\ T(\lfloor n/2 \rfloor + T(\lceil n/2 \rceil) & n > 2 \end{cases}$$

این راه‌حل چه‌گونه است؟

برای $n = 2^k$ بهینه است.

$$T(2^k) = 2[3 \times 2^{k-2} - 2] - 2 = 3 \times 2^{k-1} - 2$$

ولی در حالت کلی بهینه نیست! $T(6) = 8$ در حالی که ۷ مقایسه ($\lceil \frac{18}{2} \rceil - 2$) کافی است.

راه حل بهینه:

۱) تقسیم آرایه به ۲ و $n-2$ عنصر

۲) پیدا کردن کوچکترین و بزرگترین عنصرهای هر قسمت با $T(n-2)+1$ مقایسه

۳) ۲ مقایسه‌ی دیگر

$$T(n) = \begin{cases} 0 & n = 1 \\ 1 & n = 2 \\ T(n-2) + 3 & n > 2 \end{cases}$$

که جواب آن همان مقدار بهینه است. چرا؟

تکرار با جای‌گذاری

مثال:

$$T(n) = 3T(\lfloor \frac{n}{4} \rfloor) + n$$

تکرار با جای‌گذاری

مثال:

$$T(n) = 3T(\lfloor \frac{n}{4} \rfloor) + n$$

$$T(n) = 3T(\lfloor \tfrac{n}{4} \rfloor) + n$$

تکرار با جای‌گذاری

مثال:

$$T(n) = ۳T(\lfloor \frac{n}{۴} \rfloor) + n$$

$$\begin{aligned} T(n) &= ۳T(\lfloor \tfrac{n}{۴} \rfloor) + n \\ &= ۳^۲ T(\lfloor \tfrac{\lfloor \frac{n}{۴} \rfloor}{۴} \rfloor) + ۳\lfloor \tfrac{n}{۴} \rfloor + n \end{aligned}$$

تکرار با جای‌گذاری

مثال:

$$T(n) = ۳T(\lfloor \frac{n}{۴} \rfloor) + n$$

$$\begin{aligned} T(n) &= ۳T(\lfloor \tfrac{n}{۴} \rfloor) + n \\ &= ۳^۲ T(\lfloor \tfrac{\lfloor \frac{n}{۴} \rfloor}{۴} \rfloor) + ۳\lfloor \tfrac{n}{۴} \rfloor + n \\ &= ۳^۲ T(\lfloor \tfrac{n}{۴^۲} \rfloor) + ۳\lfloor \tfrac{n}{۴} \rfloor + n \\ &= \dots \\ &\le ۳^i T(\tfrac{n}{۴^i}) + n \sum_{j=۰}^{i-۱} (\frac{۳}{۴})^j \end{aligned}$$

داریم، $\frac{n}{۴^i} = ۱ \Rightarrow i = \log_۴ n$،

$$T(n) \le ۳^{\log_۴ n} * T(۱) + n \sum_{j=۰}^{\log_۴ n - ۱} (\frac{۳}{۴})^j$$

اما می‌دانیم که

$$\lim_{n \to \infty} \sum_{j=۰}^{\log_۴ n - ۱} (\frac{۳}{۴})^j = ۴ \Rightarrow C_۲ < ۴$$

و داریم، $۳^{\log_۴ n} = n^{\log_۴ ۳}$

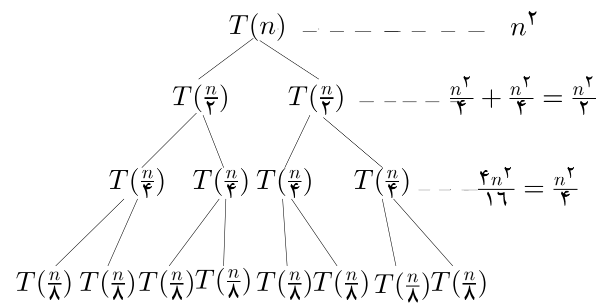
$$T(n) \le Cn^{\log_۴ ۳} + ۴n \Rightarrow T(n) = O(n)$$

مثال: $F(n) = F(n-۱) + F(n-۲)$ **و** $F(۱) = F(۲) = ۱$.

از روش حل رابطه‌های همگن استفاده می‌شود.

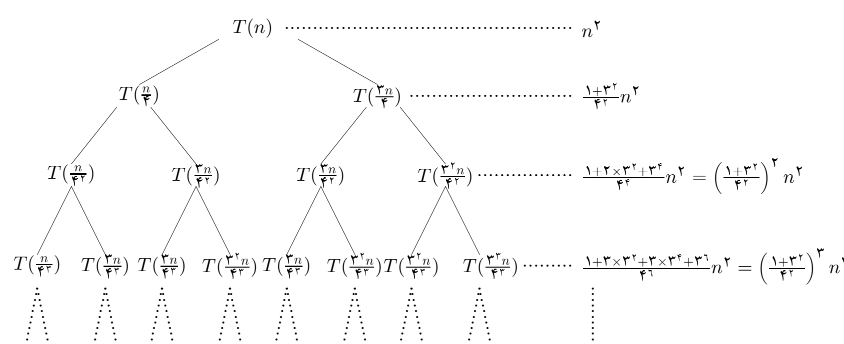
درخت بازگشت (Recursion Tree)

مثال: $T(n) = 2T(\frac{n}{2}) + n^2$



$$\begin{aligned} T(n) &= n^2 + \frac{n^2}{2} + \ldots + \frac{n^2}{2^i} + \ldots + \frac{n^2}{2^{\log n}} \\ &= n^2(1 + \frac{1}{2} + \ldots + \frac{1}{2^i} + \ldots + \frac{1}{n}) \\ &\leq 2n^2 \\ \Rightarrow T(n) &= O(n^2) \end{aligned}$$

مثال دیگر: $T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + n^2$



$$\begin{aligned} T(n) &\leq n^2 \sum_{i=0}^{\log_{\frac{4}{3}} n} \left(\frac{1+3^2}{4^2}\right)^i \\ &\leq cn^2 = O(n^2) \end{aligned}$$

قضیه‌ی اصلی

برای $a \geq 1, b > 1$ **و تابع** $f(n)$ **حل رابطه‌ی بازگشتی** $T(n) = aT(\frac{n}{b}) + f(n)$ **(که در آ**
$\frac{n}{b}$ **می‌تواند به صورت کف یا سقف باشد) به قرار زیر است:**

الف - اگر $f(n) = O(n^{\log_b a - e})$**، برای** $e > 0$ **(یعنی رشد تابع** $f(n)$ **از تابع** $n^{\log_2 a}$ **بـ**
صورت چند جمله‌ای کمتر باشد،) در این صورت، $T(n) = \Theta(n^{\log_b a})$

ب - اگر $f(n) = \Theta(n^{\log_b a})$**، دراین صورت،** $T(n) = \Theta(n^{\log_b a} \log_2 n)$

ج - اگر $f(n) = \Omega(n^{\log_b a + e})$**، دراین صورت،** $T(n) = \Theta(f(n))$.

اگر G درجه‌ی رشد تابع $g(n)=n^{\log_b a}$ و F درجه‌ی رشد تابع $f(n)$ باشد، خواهیم داشت:

۱) اگر $F > G$، $T(n)=\Theta(f(n))$

۲) اگر $G > F$، $T(n)=\Theta(g(n))$

۳) اگر $F = G$، $T(n)=\Theta(g(n)\lg n)$.

مثال: $T(n)=9T(\frac{n}{3})+n$

حل:

داریم، $f(n)=n$ **و** $g(n)=n^{\log_3 9}=n^2$.

بدیهی است که درجه‌ی رشد n^2 **از** n **به‌صورت چند جمله‌ای بیشتر است. لذا،**

$$f(n)=O(n^{2-1}) \Rightarrow T(n)=\Theta(n^2)$$

مثال: $T(n)=T(\frac{2n}{3})+1$

حل: داریم، $g(n)=n^{\log_{\frac{3}{2}} 1}=n^0=1$ **و** $f(n)=1$.

پس، $T(n)=\Theta(\log_2 n)$.

مثال: $T(n)=3T(\frac{n}{4})+n\lg n$

حل:

چون، $f(n)=n\lg n$ **و** $g(n)=n^{\log_4 3}=n^{0.793}$ **و** $f(n)=\Omega(n^{0.793+0.1})$،

داریم $T(n)=\Theta(n\lg n)$.

مثال: $T(n) = ۲T(\frac{n}{۲}) + n\lg n$

حل:

$g(n) = n^{\lg ۲} = n$ و $f(n) = n\lg n = (n)$.

اختلاف به‌صورت نمایی نیست یعنی هیچ ϵای نمی‌توان پیدا کرد که برای کلیه‌ی nهای بزرگ، رابطه‌ی $n\lg n < n^{۱+\epsilon}$ برقرار باشد.

بنابراین این مسئله را باید از روش دیگری، مثلاً استقرا حل کرد.

حدس: جواب $T(n) = O(n\lg^{۲} n)$ است که به‌صورت زیر با استقرا اثبات می‌شود.

پایه: $T(۲) \le ۲c$

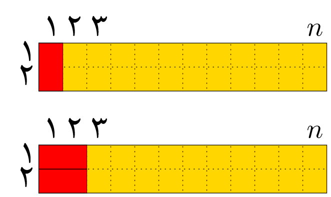
فرض: $T(k) \le ck\lg^{۲} k$ برای $k < n$.

حکم:

$$\begin{aligned} T(n) &\le ۲c\frac{n}{۲}\lg^{۲}\frac{n}{۲} + n\lg n \\ &\le cn(\lg n - ۱)^{۲} + n\lg n \\ &\le cn\lg^{۲} n + n[c + (۱-۲c)\lg n] \end{aligned}$$

برای $c = ۱$ و $n > ۲$ داریم $c + (۱-۲c)\lg n < ۰$ و حکم اثبات می‌شود.

روابط بازگشتی همگن



به چند طریق می‌توان صفحه‌ای $n \times ۲$ را با موزاییک‌های ۱×۲ فرش کرد؟

حل: اگر جدول $n \times ۲$ را بتوان با f_n روش مختلف فرش کرد، داریم:

$$f_n = f_{n-۱} + f_{n-۲}$$

و $f_۰ = ۰$ و $f_۱ = ۱$ (دنباله‌ی فیبوناچی).

تمرین: به چند طریق می‌توان صفحه‌ای $n \times ۳$ را با موزاییک‌های ۲×۱ فرش کرد؟

روابط بازگشتی همگن

اگر c_iهاعددهای حقیقی باشند، به رابطه‌ی بازگشتی زیر رابطه‌ی بازگشتی همگن ا
درجه‌ی k می‌گویند:

$$a_n = c_۱ a_{n-۱} + c_۲ a_{n-۲} + \cdots + c_k a_{n-k}$$

تابع $g(n)$ را یک جواب دنباله‌ی بازگشتی فوق می‌نامند، اگر دنباله‌ی $_n = g(n)$
$(n \in \mathcal{N})$ در رابطه‌ی بازگشتی صدق کند.

قضیه: اگر $a_n = g_i(n)$ برای $(i = ۱, \ldots, r)$ جوابی برای

$$a_n = c_۱ a_{n-۱} + c_۲ a_{n-۲} + \cdots + c_k a_{n-k}$$

باشند، آن‌گاه هر ترکیب خطی از این r جواب به صورت $A_۱ g_۱(n) + A_۲ g_۲(n) + \cdots + A_r g_r(n)$ که در آن A_iها اعدادی حقیقی‌اند، پاسخی برای رابطه‌ی بازگشتی است.

اثبات:

- **فرض کنیم،** $h(n) = A_۱ g_۱(n) + A_۲ g_۲(n) + \cdots + A_r g_r(n)$،
- **چون** $g_i(n)$ **یک جواب است، پس داریم:**

$$g_i(n) = c_۱ g_i(n-۱) + c_۲ g_i(n-۲) + \cdots + c_k g_i(n-k)$$

- **پس نتیجه می‌شود:**

$$h(n) = c_۱ h(n-۱) + c_۲ h(n-۲) + \cdots + c_k h(n-k)$$

بنابراین حکم قضیه ثابت است.

روش حل رابطه‌ی بازگشتی همگن

اگر $g(n) = x^n$، جواب رابطه‌ی بازگشتی همگن باشد، داریم:

$$x^n - c_1 x^{n-1} - c_2 x^{n-2} - \cdots - c_k x^{n-k} = 0$$

یا به عبارت دیگر،

$$x^k - c_1 x^{k-1} - \cdots - c_k = 0$$

یعنی x جواب معادله درجه k فوق است.

این معادله رامعادله‌ی مشخصه‌ی (characteristic equation) رابطه‌ی بازگشتی می نامیم

اگر x_i، ریشه‌ی معادله‌ی مشخصه باشد، بدیهی است که $a_n = x_i^n$ یک جواب رابطه‌ی بازگشتی است

بنا بر قضیه‌ی قبلی هر ترکیب خطی از x_i^nها هم یک جواب رابطه‌ی بازگشتی است.

به طور مثال ترکیب خطی:

$$a_n = t_1 x_1^n + t_2 x_2^n + \cdots + t_k x_k^n$$

در این رابطه‌ی بازگشتی باید مقادیر k عنصر اول این دنباله داده شده باشند، پس:

$$\begin{cases} a_0 = t_1 + t_2 + \cdots + t_k \\ a_1 = t_1 x_1 + t_2 x_2 + \cdots + t_k x_k \\ \vdots \\ a_{k-1} = t_1 x_1^{k-1} + t_2 x_2^{k-1} + \cdots + t_k x_k^{k-1} \end{cases}$$

k معادله و k مجهول می‌باشد.(مجهول‌ها t_1 تا t_k هستند.)

اگر x_iها متمایز باشند این دستگاه معادلات یک جواب منحصربه‌فرد دارد، یعنی بادادن k عنصر اول دنباله،میتوان جوابی منحصربه‌فرد برای دنباله پیدا کرد.

مثال: دنباله‌ی فیبوناچی را حل کنید.

حل: معادله‌ی مشخصه: $x^2 - x - 1 = 0$

ریشه‌های آن $x_1 = \frac{1+\sqrt{5}}{2}$ و $x_2 = \frac{1-\sqrt{5}}{2}$

$$f_n = t_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + t_2 \left(\frac{1-\sqrt{5}}{2}\right)^n$$

و با توجه به مقادیر اولیه داریم:

$$\begin{cases} t_1 + t_2 = f_0 = 0 \\ t_1(\frac{1+\sqrt{5}}{2}) + t_2(\frac{1-\sqrt{5}}{2}) = f_1 = 1 \end{cases}$$

و از این معادله‌ها نتیجه میشود:

$$t_1 = \frac{1}{\sqrt{5}}, t_2 = -\frac{1}{\sqrt{5}}$$

$$f_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right) \qquad (1)$$

قابل توجه است، جمله‌ی $(\frac{1-\sqrt{5}}{2})^n$ بابزرگتر شدن n بسیار کوچک می‌شود و با توجه به اینکه f_n عددی حسابی است، اگر $\langle x \rangle$ را نزدیکترین عدد صحیح به x تعریف کنیم، داریم:

$$\langle x \rangle = \lfloor x + \frac{1}{2} \rfloor$$

و با این تعریف:

$$f_n = \langle \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \rangle$$

- اگر x_i ریشه‌ی مضاعف درجه‌ی ۲ معادله‌ی مشخصه باشد، $a_n = nx_i^n$ نیز یک جواب رابطه‌ی بازگشتی است (اثبات با مشتق‌گیری از معادله‌ی مشخصه است، به این وسیل که ریشه‌ی مضاعف، ریشه‌ی مشتق معادله‌ی مشخصه است.)
- اگر x_i ریشه‌ی مضاعف درجه ۳ باشد، $n^2x_i^n$ نیز یک جواب رابطه‌ی بازگشتی است.
- در حالت کلی اگر x_i ریشه‌ی مضاعف درجه p باشد،

$$g(n) = t_0x^n + t_1nx^n + t_2n^2x^n + \cdots + t_{p-1}n^{p-1}x^n$$

جوابی برای رابطه‌ی بازگشتی است.

مثال: رابطه‌ی بازگشتی زیر را حل کنید:

$$a_n = 5a_{n-1} - 8a_{n-2} + 4a_{n-3} \ (n \geq 3)$$

و داریم: $a_0 = 0$، $a_1 = 1$، و $a_2 = 2$.

حل:

معادله مشخصه:

$$x^3 - 5x^2 + 8x - 4 = 0 = (x-1)(x-2)^2$$

بنابراین جواب کلی به این صورت است: $a_n = c_1 1^n + c_2 2^n + c_3 n 2^n$

که با اعمال مقادیر اولیه داریم

$$\begin{aligned} c_1 + c_2 &= 0 \quad & n = 0 \\ c_1 + 2c_2 + 2c_3 &= 1 \quad & n = 1 \\ c_1 + 4c_2 + 8c_3 &= 2 \quad & n = 2 \end{aligned}$$

که جواب آن $c_1 = -2$، $c_2 = 2$، $c_3 = -1/2$ است. پس

$$a_n = 2^{n+1} - n2^{n-1} - 2$$